

The Tiger Shark File System

Roger L. Haskin
Frank B. Schmuck

IBM Almaden Research Center
650 Harry Road
San Jose, CA. 95120

Abstract

Tiger Shark is a parallel file system for IBM's AIX operating system. It is designed to support interactive multimedia, particularly large-scale systems such as interactive television (ITV). Tiger Shark scales across the entire RS/6000 product line, from small desktop machines to the SP-2 parallel supercomputer.

Tiger Shark's primary features are support for continuous-time data, scalability, high availability, and manageability, all of which are crucial in its role in large-scale video servers. Interestingly, most of the features that make Tiger Shark a good video server are important for other large-scale applications such as technical computing, data mining, digital library, and scalable network file servers.

This paper briefly describes Tiger Shark: the environment that makes it important, the key technology it embodies, and the efforts to build products based on it.

Introduction

Over the past several years, radical changes have occurred in the underlying technology of computing and communication systems. The plummeting cost of computer hardware and storage, the increasing capacity of wide-area communications, and particularly the advances in data compression technology, have enabled a variety of exciting new applications.

One such application is "interactive multimedia", which promises a radical shift in the way entertainment, instruction, news, advertising, and many other forms of information are delivered to the public.

Literally, "multimedia" is the mixing of multiple data types (e.g. text and graphics). In practice, multimedia has come to imply the presentation of continuous-time data, primarily audio, video, and animation.

Network Multimedia

Interactive multimedia applications have become increasingly important in the PC market in the last several years. To date, most multimedia applications run on stand-alone PCs, with digitized video and audio coming from local hard disks and CD-ROMs. Increasingly, there has been a demand for file servers that support multimedia data. The reasons for this are identical to the ones that motivate the use of file servers for conventional data: sharing, security, and centralized administration. The initial interest in multimedia servers was in a LAN environment, but with the explosion of the Worldwide Web, there is increasing interest in presenting multimedia over a WAN.

Interactive Television

Multimedia technology allows moving television away from a one-size-fits-all broadcast paradigm to one in which all information is presented to the consumer under his interactive control.

Much hype has surrounded ITV, and there is still considerable debate about the rate at which this technology will be introduced and what form it will take. However, at the heart of any ITV system is a server sending digitized video over a large-scale distribution network to thousands of simultaneous users.

Video Servers

Conventional file servers are not very good at managing multimedia data. When a conventional file server becomes overloaded, all users see lower throughput and higher response time. For multimedia, this results in annoying jitter in the presentation. This can be circumvented by under-loading a conventional file server, but handling multimedia in a cost-effective manner requires a true “video server” with special support for *continuous-time data*.

In ITV, providing continuous-time access is exacerbated by the sheer magnitude of the system. Potential customers for ITV are calling for servers that support tens of thousands of simultaneous viewers. A single video stream requires between 1.5 and 6 Mbits/sec of bandwidth. Even a 1000-stream video server for 6 Mb/sec video streams (typical of what is required for high-end ITV) must have a throughput of 750 megabytes per second. Compare this to most conventional network file servers, which run at well under 10 MB/sec. On an SP-2, a 1000-stream server requires at least 38 nodes (each with 20 MB/sec throughput)¹, all accessing the same video data, making the need for *scalability* in an ITV server obvious.

ITV is a line-of-business application, making the need for *high availability* and *manageability* obvious as well. A 10,000-stream ITV system presenting 2-hour movies at \$5 each generates \$25,000/hour. Routine component failures must not take down the server or even unduly interrupt viewers, and it must also be possible to repair and reconfigure the server while it remains operational.

The Tiger Shark File System

Broadly speaking, a video server consists of three components: a *control* component that responds to client requests, a *communication* component that moves data across the network connection between the client and server, and a *file system* component that manages the storage and retrieval of data from disk. The native file system on AIX, the *Journalled File System* (JFS) meets none of the stated requirements for use in a video server. To enable the use of the RS/6000 and SP platforms as video servers, we developed the Tiger Shark file system, which was designed specifically to meet the requirements of a video server:

1. This limit is imposed by the microchannel on existing SP nodes.

- Tiger Shark contains designed-in support for continuous-time data: *admission control* to prevent overload, and *disk scheduling* to assure data is read on time.
- Tiger Shark supports effectively unlimited scalability both in the amount of data it can store and the data throughput (bandwidth) it supports.
- Tiger Shark is designed for high availability - it remains operational in the face of any single disk and/or node failure.²
- Tiger Shark is largely self-managing - automatic load balancing, for example, is inherent in the design. All operator-initiated management functions can be performed while the system remains operational.

In addition to supporting video, Tiger Shark contains a number of features that make it suitable as a general-purpose parallel file system:

- Tiger Shark is implemented as a standard AIX Virtual File System, and presents a Posix-compliant programming interface, so applications that use it require little or no porting.
- Tiger Shark supports high-speed access to a file from a single application or from any number of applications running in parallel.
- Tiger Shark is fully cache-coherent across nodes in the SP. It is designed to support byte-range locking, allowing parallel access to non-overlapping regions of a file with no communication overhead.

In summary, Tiger Shark enables the RS/6000 and the SP to efficiently support the data access demands of both multimedia and parallel computing.

Tiger Shark Overview

The Tiger Shark file system runs on a cluster of processors (*file system nodes*) that share a pool of disks. Each file system node is assumed to be able to access all disks. A single RS/6000 can be considered as a degenerate case of such a cluster. In the SP-2, a software component called *VSD* (Virtual Shared Disk) allows file system nodes to use the high-speed switch to access disks that are physically attached to

2. This is true when Tiger Shark is properly configured and running on a multi-node machine (e.g. an SP-2), and when the underlying hardware itself has no single failure point (e.g. dual power). Tiger Shark can be configured to survive multiple failures, as long as successive failures occur far enough apart in time to allow recovery.

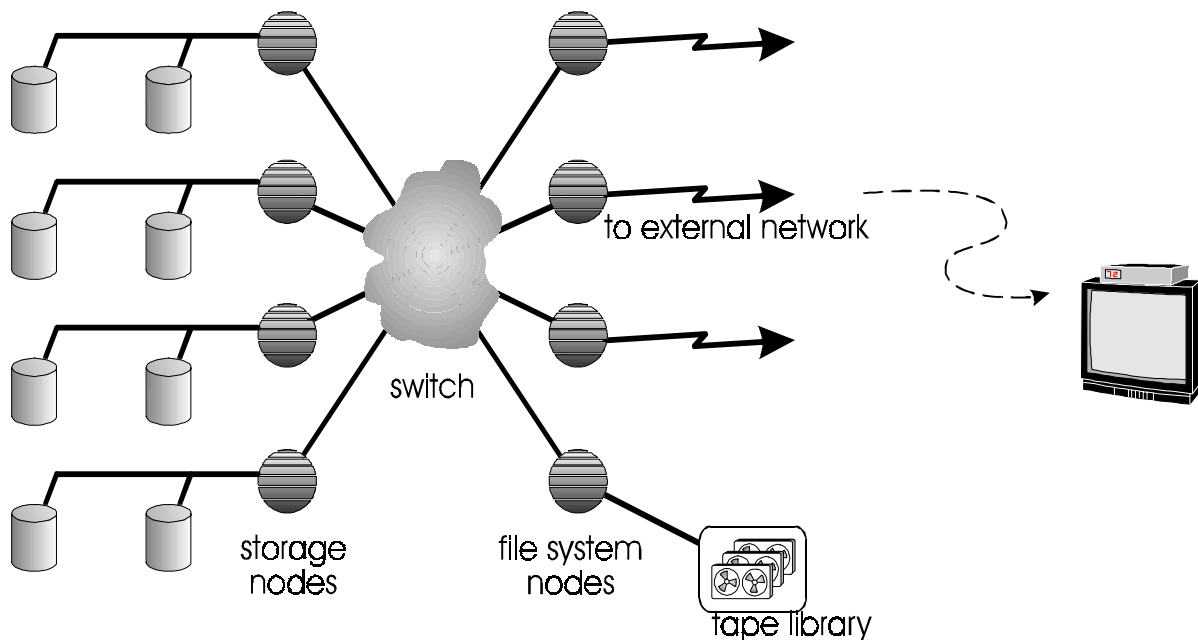


Figure 1. Tiger Shark on SP-2

storage nodes.³ This is illustrated in Figure 1. In addition to supporting single RS/6000s and the SP-2, a cluster can consist of multiple RS/6000s with SSA⁴ disks. The current generation of SSA disks on the RS/6000 limits such an “SSA Cluster” to 16 SSA loops and 8 RS/6000 nodes. Each node is connected to all loops, so any node can directly access all disks. The SSA cluster allows a very cost-effective server within its scaling limits.

Tiger Shark makes files available to applications, including network services such as the NFS (Network File System) daemon, using the AIX VFS (Virtual File System) interface. The VFS interface makes Tiger Shark compatible with the native JFS file system; programs don’t have to be modified to use Tiger Shark unless they want to take advantage of its special support for multimedia.

Tiger Shark presents a single-system image to its clients — operations on the same file from more than one file system node see a globally-consistent view of the file. In its primary role as a video server, the cluster is connected to an external network (ATM, Switched Ethernet, etc.) through

3. A single node can serve as both a file system and storage node, but to keep the discussion simple, we will always treat storage and file system nodes as though they were separate.

4. Serial Storage Architecture.

network interfaces attached to file system nodes. From Tiger Shark’s point of view, programs that makes files accessible to external clients (e.g. the NFS daemon) are simply applications running on the file system nodes. Another such application might be a tape archive manager that retrieves less frequently used files from a tape library attached to one of the file system nodes.

Tiger Shark Architecture

Tiger Shark has a number of architectural elements that allow it to meet its design goals. Following is a brief discussion of the major ones:

Support for Continuous-Time File Access. There are two aspects to support for continuous-time: resource reservation and disk scheduling. Resource reservation ensures that there is sufficient disk throughput to satisfy existing clients; new requests are rejected if not. Real-time scheduling executes disk I/Os in the proper order to achieve an uninterrupted flow of data to clients. Tiger Shark resource reservation uses a “benchmarking” process that measures the I/O system against a synthetic workload designed to approximate multimedia access. It automatically detects and compensates for I/O bottlenecks such as disk adapters, I/O busses, etc. Tiger Shark uses deadline scheduling (as opposed to conventional scan or elevator algorithms). Tiger

Shark deadline scheduling can achieve over 90% of maximum disk throughput for multimedia traffic while maintaining full quality of service.

Large Disk Blocks. Efficiently supporting multimedia and supercomputing requires a file system to maximize disk throughput. Conventional file systems optimize the use of disk space by using a small (4 Kbyte) disk block size. Since disk throughput is strongly related to disk block size, Tiger Shark uses a much larger (256 Kbyte) block size. Because multimedia and supercomputing usually use very large files, the impact of large blocks on disk storage utilization is typically negligible.

Wide Striping. In both supercomputing and multimedia, much or all file system activity is often directed at a single file.⁵ The throughput of an individual disk is only about 5-10 MB/sec, so achieving higher throughput than this from a file requires spreading it across multiple disks. Each Tiger Shark file system can reside on as many as 65536 disks, allowing effectively unlimited throughput. Since each file is striped evenly across all disks in a file system, load against the disks is inherently balanced regardless of file access patterns.

Replication. As the number of disks and nodes in the system scales, the probability of individual component failures increases. Some form of data redundancy is required for the system to continue to operate in the face of component failures. RAID is only a partial solution. Commercially available RAID subsystems can be over twice as expensive as conventional disk, RAID controllers are often performance bottlenecks, and for a large video server (commonly 1TB of disk storage) the probability that an entire RAID subsystem will fail is significant. As an alternative to RAID, Tiger Shark supports block-level replication of both file system data and the bookkeeping information (metadata) that keeps track of the location of files on disk.

On-line System Management. Tiger Shark responds to signals generated by the AIX operating system to automatically recover and reconfigure in response to the failure or repair of hardware components. Reconfiguration occurs while the system remains operational. System administrator initiated functions (creating and deleting file systems, adding and removing disks from file systems, etc.) also are executed while the system remains operational. Files can

5. An example is an ITV system the first night *Terminator 2* goes on-line; many or all clients may be viewing this one movie. For a 1000 stream server, this would be as much as 750 Mbytes/sec. from the file containing *Terminator 2*.

be reorganized (e.g. to restripe them onto newly-added disks) on-line, with the reorganization taking place in the background.

Disk Data Structures

Tiger Shark supports multiple, separately mountable file systems, each stored on a dedicated group of disks called a *stripe group*. Each file in a file system is evenly distributed (or *striped*) across all disks in the stripe group.

Tiger Shark file system data structures are similar to those of a conventional file system. Each file has a fixed-size *inode*, which contains information such as the file size and time of last modification. The inodes point to *indirect blocks*, which themselves point to the file's data blocks. To efficiently use disk space, data blocks can be split into *fragments* (fractional blocks), which efficiently store small files as well as the last piece of large files. Unlike a traditional Unix file system, any object (inode, indirect block, data block, etc.) can be replicated. The table used to map disk numbers to physical disk names also contains information that allows Tiger Shark to allocate replicas on disks with no common failure point. The degree of replication can be controlled on a per object basis, for example replicating only metadata, or only the most popular movies. This allows trade-offs to be made between fault tolerance and system cost.

Software Structure

Figure 2 illustrates the Tiger Shark software structure. Most of Tiger Shark is implemented as a user level daemon process (labeled "Tiger Shark File System Daemon" in Figure 3) running at real-time priority under AIX. The VFS interface is implemented by a dynamically loaded kernel extension. The client application accesses Tiger Shark files through standard AIX file system interface (open, close, read, write, etc.). The VFS kernel extension interacts with the Tiger Shark daemon to perform the requested file system operation.

The VFS kernel extension uses AIX message queues to communicate with the Tiger Shark daemon. File data is transferred through a memory segment shared by the Tiger Shark daemon and the VFS layer. Once a data block has been read into a buffer in shared memory, the VFS layer can satisfy read requests directly from the buffer without sending a message to the daemon. For example, if a client program issues sequential 4k reads to a Tiger Shark file, the

VFS layer sends one message to the daemon for each 256K block (i.e. one for every 64 read calls).

For continuous-time applications Tiger Shark provides extensions to the normal file system interface that allow the client to specify quality of service requirements such as the video playback data rate. Tiger Shark uses this call to reserve resources (disk bandwidth, switch capacity, buffer space, etc.).

Tiger Shark can be accessed through the standard AIX NFS daemon to allow PC and workstation clients to play multimedia (audio and video) files over a LAN⁶. In this application, both the NFS client code and the AIX NFS server code are unmodified, and are therefore not aware of the Tiger Shark API extensions for quality of service. To support multimedia playback through NFS, Tiger Shark contains *streaming heuristic* code that allows a *default playback rate* to be assigned to a file. When Tiger Shark detects an NFS client performing sequential reads from a file with a default playback rate, Tiger Shark assumes the client is streaming, and a) reserves bandwidth for the client and b) uses deadline scheduling to read the file at the default rate⁷.

Tiger Shark Applications

Multimedia

Tiger Shark has to date been used in three IBM ITV trials as well as several customer betas and joint studies. Following is a short overview of some of these activities.

The Bell Atlantic Field Trial ([PL93], [WN93]) was one of the first video on demand (VOD) trials to use a video server (some earlier trials used banks of VCRs to simulate a video server). It featured Shark ([HASK93]), a predecessor of Tiger Shark, supporting up to 50 simultaneous streams from a single RS/6000 Model 970. The video was sent via T1 to set-top boxes located in the homes of approximately 400

6. This LAN must have adequate bandwidth and low enough response time to support video, e.g. switched ethernet or ATM LAN emulation.

7. It should be noted that the default playback rate is the maximum rate that Tiger Shark will read ahead during streaming. If the client attempts to read at a higher rate, it will be blocked. If the client reads at a lower rate, read-ahead is limited to a per-client maximum number of buffers.

subscribers using ADSL modems for the subscriber loop. This system was truly VOD, in that it featured movies that were sent to individual households on demand, but was not interactive in that movies were requested via an automated telephone order entry system (Direct Talk/6000). The Bell Atlantic Field Trial became operational in April 1993, and continued through September 1994.

The Hong Kong Telecom VOD Trial ([PAT94], [HASK95]) supported 150 simultaneous 1.5 Mbit/sec. streams from two RS/6000 Model 980 servers running Tiger Shark. Video distribution was via T1/ADSL to set-top boxes that also featured an X.25 back-channel for sending interactive control input to the server. The user interface featured interactive menus with image overlays and video fly-ins. These came from Tiger Shark via the T1 channel as well, requiring the server to support true interactive response time. The trial ran successfully from March through September of 1995.

The Tokyo Metropolitan Government ITV Trial ([TMG95]) is scheduled to begin in early 1996. It will feature 100 6 Mbit/sec. streams from Tiger Shark running on an SP-2. Video delivery is via hybrid fiber/coax.

Argonne Labs, one of the national supercomputing centers, and IBM collaborated to develop an experimental WAN-based video server employing Tiger Shark on a 28-node SP-2. This system uses an implementation of RTP (Real-Time Protocol) to transmit video over the M-Bone (multicast backbone) network interconnecting the supercomputing centers. The system was demonstrated at Supercomputing '95 conference in San Diego (Dec. 3-6, 1995) with video running to an IBM location in New York ([IBM95]).

Technical and Commercial Applications

Tiger Shark's attributes of scalability, availability, manageability, and a standard programming interface are important in many technical and commercial applications. This has resulted in considerable interest in Tiger Shark as a general purpose parallel file system.

To better support general purpose parallel computing, several extensions to Tiger Shark are underway. The most significant is support for *fine-grained write sharing* of individual files. Video serving is primarily read-only, but a large simulation can involve many nodes simultaneously

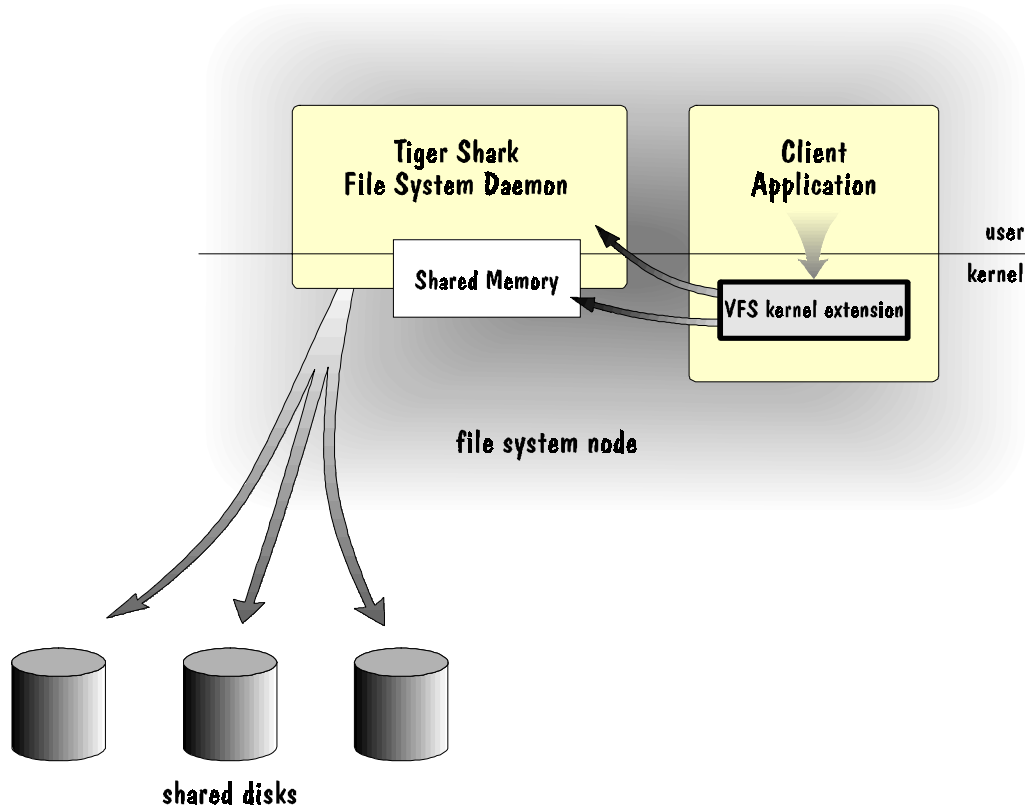


Figure 2. Software Architecture Overview

making updates to small pieces of a file on which they are individually working.

Summary

Multimedia applications place demands on the file system beyond the ability to support quality of service. Tiger Shark not only supports the real-time access required for multimedia, but provides a scalable, robust platform adequate for large ITV systems. Tiger Shark has proven itself in a number of real-world customer trials, and is being extended to support more general-purpose parallel computing.

References

[HASK93] R. Haskin, *The Shark Continuous Media File Server*, Proceedings of IEEE 1993 Spring COMPCON, San Francisco, CA, Feb. 1993, pp. 12-17.

[HASK95] R. Haskin and F. Stein, *A System for the Delivery of Interactive Television Programming*, Proceedings of IEEE 1995 Spring COMPCON, San Francisco, CA, Mar. 1995, pp. 209-216.

[IBM95] *Argonne National Lab Video Server* <http://lscftp.kgn.ibm.com/pps/vibm/show/super95/demos/video.html>

[PAT94] Alan Patterson, *Hong Kong Telecom, IBM Map Video Effort*, *Electronic Engineering Times*, Aug. 1, 1994, p. 20.

[PL93] Larry Plumb, *Bell Atlantic Demonstrates Video on Demand over Existing Telephone Network*, Bell Atlantic press release, Jun. 14, 1993.

[TMG95] *Basic Plan for the Multimedia Experiments at the Tokyo Metropolitan Waterfront Subcenter Multimedia Experiments*, <http://www.tokyo-teleport.co.jp/english/atms/atmsj201.htm>

[WN93] *Bell Atlantic, IBM Announce Agreement for Video on Demand Server*, *World News Today*, Jan 8, 1993.